Page G Plan

Remote access to local console applications

Mar 10, 1992

The local station console interface is simple and can be emulated from any host platform. It allows remote emulation of many page applications already written without replicating their functionality on the host. It is already supported from a Macintosh, workstation, Vax, or another local station console. This note describes the logic used to facilitate implementation on other platforms.

The local console display consists of 16 lines of 32 characters each. These 512 characters are kept in an image buffer starting at address $000400 in the target station. The first 32 characters are the top line, the next 32 characters are the second line, etc. Alpha betic characters are upper case only; lower case characters need not apply. In addition, the sign bit of each character byte signifies that the character is dis played in inverse video.

The basic scheme for capturing the display is simply to make a periodic request for the 512 bytes of image buffer memory. Any characters which change are then updated on the host's display or window. The cursor location can be monitored by the memory word at $000F7C for the column# 0–31 and the word at $000F7E for the row# 0–15. The host display may update to show this cursor position on its own display, if needed. Writing to these two words can change the cursor position to allow a mouse click to set the cursor location, for example.

Typing on the screen is supported by sending the character code to memory address $000F69 using a one-byte memory setting. Note that the sign bit of the new character *must be set* to announce to the local station system that the character is new. The following character codes are used:

$1E	é¾	Home (top line, left column)
$0B	↑	Up arrow
$0A	↓	Down arrow
$08	←	Left arrow
$0C	→	Right arrow
$0D		Return (to start of current line)
$1B	ESC	Keyboard interrupt

To each code add $80 to produce the byte value used in the setting.

The state of the push-button lites is given by the two bytes at $000F78 (units) and $000F79 (modes). The bit assignments of these lites are as follows:

Bit#	*Units*	*Bit#*	*Modes*
7	(unused)	7	(unused)
6	Eng	6	A/D
5	Volts	5	D/A
4	Hex	4	Nom
3	Raise	3	Tol
2	Lower	2	Set
1	Left-right	1	Keyswitch
0	Up-down	0	Keyboard int

To make a change in the lites selection, note that there are two mutually exclusive groups of lites, the A/D-D/A-Nom-Tol-Set group and the Eng-Volts-Hex group. Selection of any lite in a group means that the other lites in that group should be turned off. The local station only updates these latching mutually-selectable lites when a push-button is pressed, leaving these bits open to remote access.

The other lites are DC. Turning on a given bit turns on the lite; turning off the bit turns off the lite. The page applications monitor the state of the *lites* to know what to do. With the local console in place, pressing the switches is one way to control these lites; settings from a remote host to the proper lites byte is another.

Since these other lites *are* DC, however, the local station updates the lites to reflect the current state of the switches every cycle. This makes it hard for a remote host to confidently change the lites between the last update and the page application's monitoring of it. A solution to this problem lies in recognizing that the hi bit of each lites byte is unused, due to the hardware interface design. When a remote host makes a setting to one of these bytes, it should also set the hi bit to signal that it is new from the host, much like the keyboard character logic. When the local station processing is about to update the lites byte, it should check the hi bit of the present lites byte value; if it is set, merely clear that bit, else update the lites byte normally. This gives a chance for the page application to notice the remote setting, but it also builds in an automatic reversion to maintaining the lites according to the current console switch states.

Knob control is not at present supported, due to the way it was originally implemented. Every cycle, the latest reading of the knob counter from the console interface is monitored for changes. If a remote setting changed the value of this byte, it would only be overwritten the next cycle, at least if there is a local console attached. Without a local console, manipulation of the knob byte reading could effect knob control. If it is important to implement knob control, the system should be changed to work with knob count differences. Then the remote host could write to this difference count byte, which would be cleared when used.

In summary, the data acquisition is to periodically collect the 512 bytes of image buffer memory, and also the 8 bytes from $000F78–000F7F, to obtain the lites bytes and the cursor position words. Settings are accomplished by writing to either of the two lites bytes or to the keyboard character input byte, or perhaps to the cursor position words.